Artistic Gymnastics Skill Action Recognition Using an Attention-Based RNN

Vanessa Chambers
Massachusetts Institute of Technology
vchamber@mit.edu

Abstract

I propose an attention-based recurrent neural network, taking keypoints as inputs, to perform action recognition on a specific set of gymnastics skills. In addition to gymnastics, action recognition has many real applications that have been studied previously. However, this paper proposes a novel method to classify gymnastics skills; my method first extracts keypoint features from each video frame of a given skill and uses these as inputs to the network.

1. Problem Definition

Gymnastics is a sport comprised of several different actions, referred to as "skills," with varying difficulty. The quality of a skill performed is assessed based on strict criteria outlined in the gymnastics code of points [1]. These criteria often require that specific body angles and body positions be displayed during skills. However, all gymnastics judging is performed with "naked-eye" assessments. Though judging skill quality has an objective basis, because it is difficult to determine exact angles and body measurements with the naked eye, this method of judging leads to somewhat arbitrary and inconsistent evaluation of skills.

This project aims to apply deep learning to this problem to lay the groundwork for a potentially more objective and consistent way to recognize and analyze gymnastics skill performance. In this paper, I use pose estimation to extract sequences of keypoint skeletons that model a gymnasts' motion, and these keypoints serve as inputs to the classification network. By omitting any extraneous information, such a visual appearance, the approach potentially reduces learned biases regarding the input data. The keypoints are then fed into an RNN, which then learns to classify the skill based on the sequence of keypoints.

A proposed next step for this project is to create a rules-based algorithm that can then score the performance of the identified skill based on the positions of the keypoints during the sequence.

2. Related Work

I use OpenPose, a pose estimation network, to extract keypoint skeletons from each video sequence. OpenPose uses partial affinity fields (PAFs) to learn the spatial relationship between different body parts to generate a skeleton [2]. Though OpenPose is robust in estimating the pose of most motions, this network has some limitations; OpenPose assumes that bodies will be in an upright position, with feet towards the ground (pointing in the direction of gravity), and the upper body in the opposite direction. Because many gymnastics skills involve upside-down body positions, OpenPose does fail at estimating the poses of skills that involve those positions. Despite this limitation, I chose to continue using OpenPose due its high support, and because of a lack of computational resources to use a different network.

Other scholars have used a very similar approach using keypoints as inputs to RNNs to aid in action recognition. Connolly and Collaborators [4] explore this problem applied to trampoline gymnastics. Ko and collaborators [6] also use OpenPose to generate hand, face, and body keypoints to recognize sign language sequences. Their network uses an RNN architecture made up of stacked bidirectional GRUs (whereas our network uses an LSTM layer).

Additionally, I created a dataset of 10 classes for this project, with each class representing a different gymnastics skills performed on the balance beam apparatus. However, it is worth mentioning that Shao and collaborators have proposed FineGym [8], a hierarchical gymnastics skill dataset made specifically for the task of action recognition. This dataset includes more examples and proposes a more robust data source for this task. Thus, future work on this task could be improved by using this specialized dataset.

3. Proposed Method

3.1. Data Collection

To generate a dataset, I select ten different classes, with each class representing a distinct gymnastics skill on the balance beam apparatus. 100 different video clips of each skill performed are collected for each class, resulting in a to-

tal of 1000 data points in the dataset. These video sequences are then input to OpenPose, which estmates the poses across each timestep for every given example. From the OpenPose data, I extract 25 keypoints corresponding to the gymnasts' body parts; each keypoint is represented by (x, y) coordinates and a confidence score. I then use these keypoints and their corresponding labels as the dataset for the network.

I determine the number of timesteps of the longest sequence. Because the keypoint sequences are of different lengths, I use this maximum value (281 timesteps) to zero pad every other sequence to conform to this length, thus, giving every keypoint sequence a uniform length. Therefore, each skill sequence is stored in an array of shape (max_frame, num_keypoints). I use two different normalization techniques using min-max normalization: keypoint normalization and timestep normalization. The former technique involves normalizing individual keypoints across each timestep in a sequence; the latter technique involves normalizing all keypoints within a given timestep. After pre-processing, I then divide data 80/20 for train/test sets.

3.2. Network Architecture

The proposed network attempts to learn the temporal dependencies between keypoint sequences. The sequences are fed into an input layer of shape (batch_size, max_timestepes, num_keypoints). I then add an LSTM layer with 64 cells on top of the input layer, which serves to capture the temporal elements of the data. This layer is then followed by two self-attention layers, which learns the importance of specific timesteps in each sequence. To reduce overfitting, I also follow each attention layer with dropout layers, each having a dropout rate of 20%. Following the dropout layers is a fully connected layer with a ReLU activation, and the final output layer uses a softmax activation to produce class predictions for each example. 1

```
Model: "model_55"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
input_58 (InputLayer)        [(None, 281, 50)]         0         []

lstm_101 (LSTM)              (None, 281, 64)           29440     ['input_58[0][0]']

attention_38 (Attention)     (None, 281, 64)           0         ['lstm_101[0][0]',
                                                                  'lstm_101[0][0]']

dropout_89 (Dropout)         (None, 281, 64)           0         ['attention_38[0][0]']

attention_39 (Attention)     (None, 281, 64)           0         ['dropout_89[0][0]',
                                                                  'dropout_89[0][0]']

dropout_90 (Dropout)         (None, 281, 64)           0         ['attention_39[0][0]']

dense_131 (Dense)            (None, 281, 128)          8320      ['dropout_90[0][0]']

dense_132 (Dense)            (None, 281, 10)           1290      ['dense_131[0][0]']

=================================================================================================
Total params: 39,050
Trainable params: 39,050
Non-trainable params: 0
_________________________________________________________________________________________________
```

Figure 1. Model architecture summary

A batch size of 120 is used, and the model is trained for 20, and 100 epochs for both of the differently normalized datasets. The model is trained again for 500 epochs for the timestep-normalzied dataset.

I analyze the performance of this model using classification accuracy and recording the loss over epochs.

4. Results

The keypoint-normalized data achieves loss values of 1.7335 and 0.7733 after 20 and 100 epochs respectively 2.

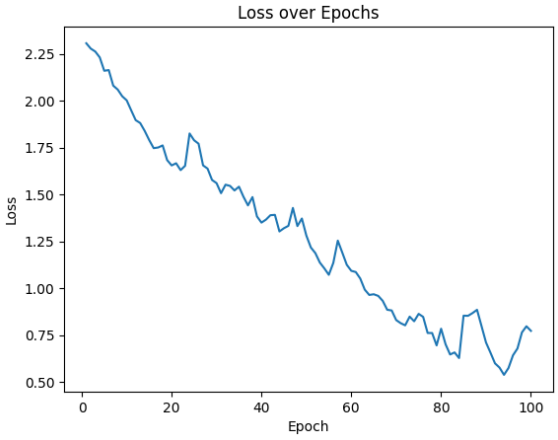

Figure 2. Loss over 100 epochs for keypoint-normalized data

Overall classifcation acccuracy is 43.22% and 63.21% after training for 20 and 100 epochs, respectively. The heatmap reflects the per-class accuracy performance after 100 epochs for this normalization strategy 5.

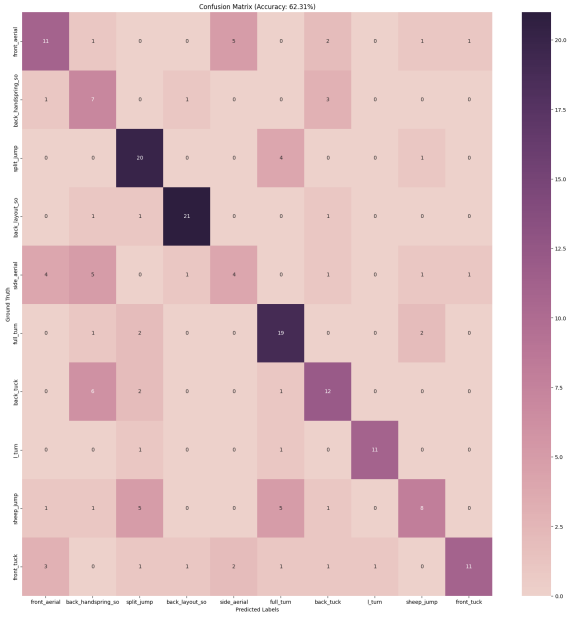

Figure 3. Classification accuracy after 100 epochs for keypoint-normalized data

The timestep-normalized data achieves loss values of 1.6786, 0.52123, and 0.0085 after 20, 100, and 500 epochs, respsectively 4.

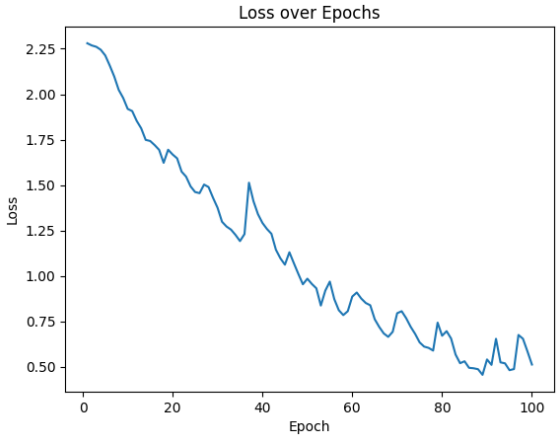

Figure 4. Loss over 100 epochs for timestep-normalized data

Overall classifcation acccuracy is 64.82%, 72.36%, 76.38% after training for 20, 100, and 500 epochs, respectively. The following heatmap reflects the per-class accuracy performance after 100 epochs for this normalization strategy 5.

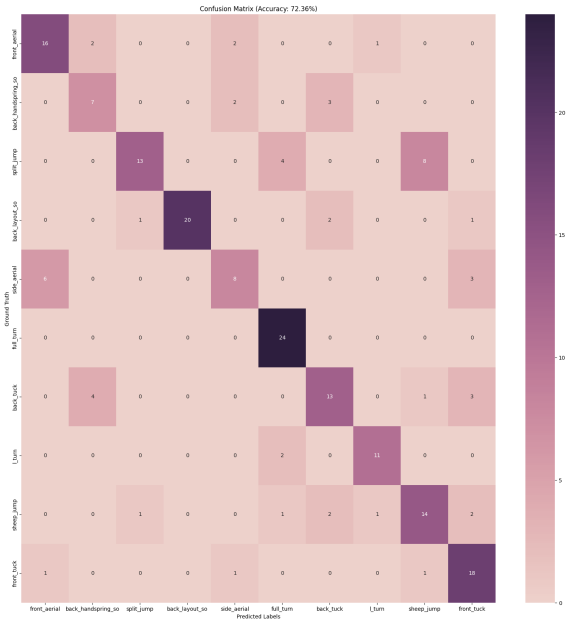

Figure 5. Classification accuracy after 100 epochs for timestep-normalized data

Loss and class accuracy for each setup is reflected here 2.

5. Conclusion

Based on the results, the timestep-based normalization pre-processing step resulted in better classification accuracy than the keypoint-based normalization. This may be caused by inaccurate normalization across timesteps; pose estimation across many timesteps could be very noisy, whereas it is more consistent within timesteps. Additionally, I normalize data after zero-padding. Therefore, the min-max normalization for keypoints across several timesteps could actually be inaccurate, as an arbitrary zero value could have been used for the minimum value, rather than the true minimum.

For the timestep-based data, the training loss reaches a minimum after $\approx$ 200 epochs. After this, the loss becomes noisy-this suggests that training for more epochs could introduce some overfitting.

Overall, for 100 epochs and after applying timestep-based normalization to the dataset, the classification accuracy is 72.36%. Classification is particularly robust for the "full_turn" and "back_layout_so" classes. Interestingly, the network seems to struggle differentiating the "front_aerial" and "side_aerial" classes–this could be attributed to the fact that these skills have very similar dynamics.

While classification is fairly accurate, there are some areas for improvement for this model. There are a few failure modes that could be contributing to the inaccuracies of this network. One failure could be attributed to noisy data; because OpenPose fails to generate accurate keypoints for upside-down bodies, any sequence involving upside-down motion in our dataset will have inaccurate input values. These visualizations of the keypoint data reflect this failure mode 6 [5].

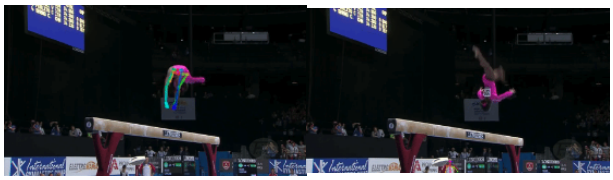

Figure 6. Pose estimation failure during upside-down motion

In addition to this, the dataset is relatively small; the lack of data could contribute to poor results. Finally, another contributor to this low accuracy could be caused by over-fitting to dominant camera angles. OpenPose extracts 2D keypoints from each video sequence; thus, keypoint sequences for the same skill are not invariant to 3D rotations. Most balance beam videos are captured from camera angles in this general region 7 [3]; because I do not have a multi-view approximation of keypoints for every sequence, the model could be over-fitting on keypoints from the more heavily used camera angles. Thus, gymnastics skills performed at different angles are potentially less likely to be recognized by the model.

Figure 7. Default camera angle for most video sequences

The robustness of this model could be improved by using a more representative dataset, for example FineGym [8]. Additionally, I could use a pose estimation network that better handles upside down body positions, such as DCPose [7]; this could generate more accurate keypoints for each class.

Finally, this project serves to lay the groundwork to achieve more objective scoring of gymnastics skills. Though, I have not yet implemented this step in this project, once I achieve more accurate classification, I could then apply a rules based scoring to each specific skill. This score system would assess the quality of the skill sequence based on the relative positions and angles of the keypoints. This could be one approach at attaining consistent and less biased scoring of gymnastics skills.

References

[1] 2022 - 2024 code of points. In *FÉDÉRATION INTERNATIONALE DE GYMNASTIQUE*, 2021. 1

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2019. 1

[3] Olympic Channel. Beam final - women's artistic gymnastics — london 2012 replays (online video). 3

[4] Paul W. Connolly, Guenole C. Silvestre, and Chris J. Bleakley. Automated identification of trampoline skills using computer vision extracted pose estimation, 2017. 1

[5] International Gymnastics Federation. 2013 artistic gymnastics world championships - women's bb and fx finals(online video). 3

[6] Sang-Ki Ko, Jae Gi Son, and Hyedong Jung. Sign language recognition with recurrent neural network using human keypoint detection. In *Proceedings of the 2018 Conference on Research in Adaptive and Convergent Systems*, page 326–328, New York, NY, USA, 2018. Association for Computing Machinery. 1

[7] Zhenguang Liu, Haoming Chen, Runyang Feng, Shuang Wu, Shouling Ji, Bailin Yang, and Xun Wang. Deep dual consecutive network for human pose estimation, 2021. 4

[8] Dian Shao, Yue Zhao, Bo Dai, and Dahua Lin. Finegym: A hierarchical video dataset for fine-grained action understanding, 2020. 1, 4

A. Results (Cont'd)

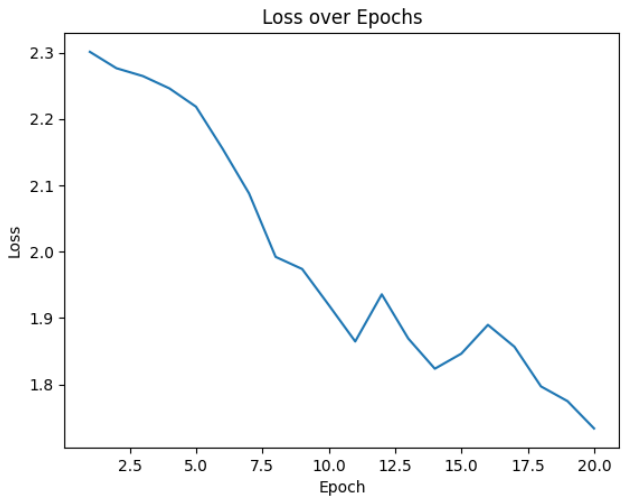

Figure 8. Loss over 20 epochs for keypoint-normalized data

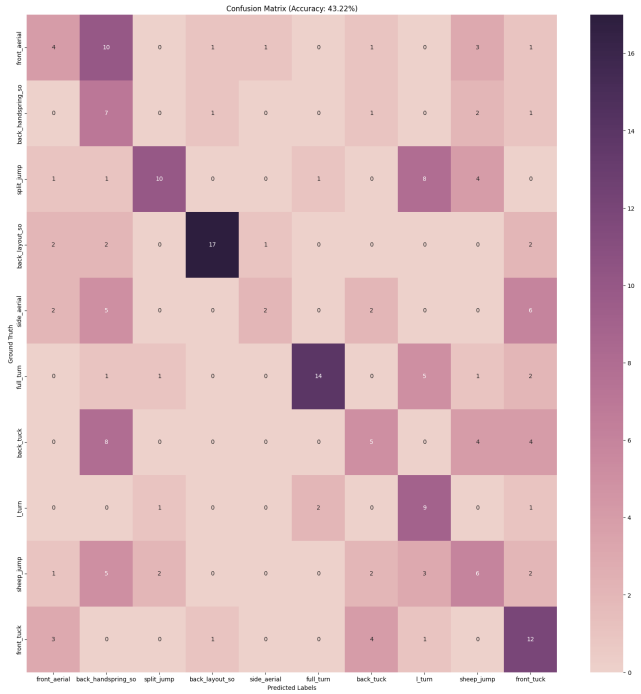

Figure 9. Classification accuracy after 20 epochs for keypoint-normalized data

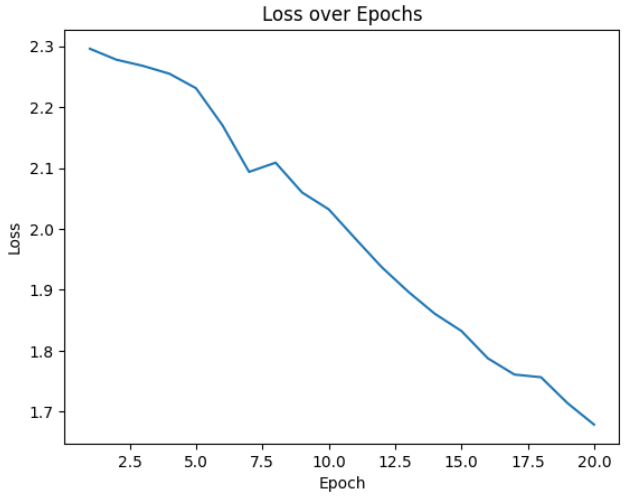

Figure 10. Loss over 20 epochs for timestep-normalized data

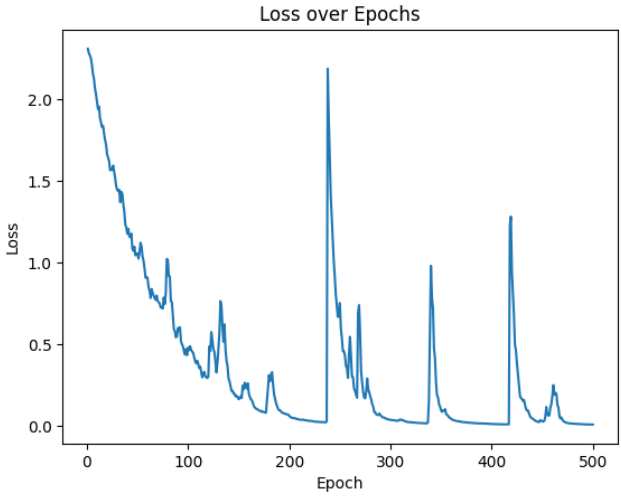

Figure 11. Loss over 500 epochs for timestep-normalized data

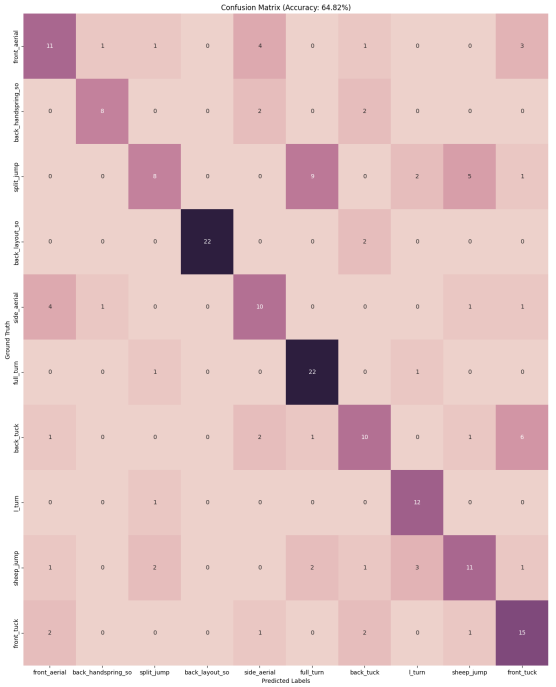

Figure 12. Classification accuracy after 20 epochs for timestep-normalized data

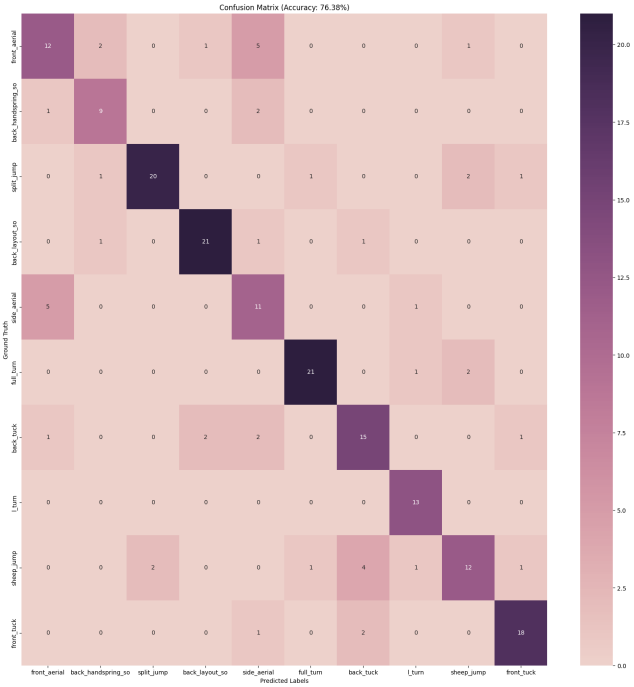

Figure 13. Classification accuracy after 500 epochs for timestep-normalized data

	Loss after 20 epochs	Loss after 100 epochs	Loss after 500 epochs
keypoint-normalized	1.7335	0.7733	
timestep-normalized	1.6786	0.52123	0.0085

Table 1. Loss for different training times.

	Accuracy after 20 epochs	Accuracy after 100 epochs	Accuracy after 500 epochs
keypoint-normalized	43.22%	63.21%	
timestep-normalized	64.82%	72.36%	76.38%

Table 2. Classification accuracy for different training times.